**Kruskal's algorithm:-**

**# define the function to find the parent of a node in the disjoint set**

**def find\_parent(parents, node):**

**if parents[node] != node:**

**parents[node] = find\_parent(parents, parents[node])**

**return parents[node]**

**# define the function to union two nodes in the disjoint set**

**def union(parents, ranks, node1, node2):**

**parent1 = find\_parent(parents, node1)**

**parent2 = find\_parent(parents, node2)**

**if parent1 == parent2:**

**return**

**if ranks[parent1] < ranks[parent2]:**

**parents[parent1] = parent2**

**elif ranks[parent1] > ranks[parent2]:**

**parents[parent2] = parent1**

**else:**

**parents[parent2] = parent1**

**ranks[parent1] += 1**

**# get the number of vertices and edges from the user**

**n = int(input("Enter the number of vertices: "))**

**m = int(input("Enter the number of edges: "))**

**# get the edges and their weights from the user**

**edges = []**

**for i in range(m):**

**u, v, w = map(int, input("Enter an edge (u v w): ").split())**

**edges.append((w, u, v))**

**edges.sort()**

**parents = list(range(n))**

**ranks = [0] \* n**

**mst = []**

**mst\_weight = 0**

**for edge in edges:**

**w, u, v = edge**

**if find\_parent(parents, u) != find\_parent(parents, v):**

**union(parents, ranks, u, v)**

**mst.append((u, v))**

**mst\_weight += w**

**print("Minimum Spanning Tree:")**

**for edge in mst:**

**print(edge)**

**print("Weight:", mst\_weight)**